Question > PDF export

Question

Background

Your platform allows **third-party developers** or **end-users** to write small JavaScript "snippets" (e.g., for analytics, UI customization, or automation). These scripts are meant to run **in the browser** — but allowing raw, unchecked JavaScript introduces serious risks:

- Use of `eval()` or `Function()` could execute malicious code.
- Top-level `await` or experimental syntax could break older browsers.
- Accessing browser globals (`window`, `document`) could leak data or tamper with your app.

You need a way to:

- Transpile modern JS/TS into browser-compatible code.
- Statistically **analyze and sanitize** the code before it hits the browser.
- Ensure code executes safely without compromising security or stability.

You're tasked with creating a backend API that transforms and secures user-provided JavaScript code so it can be safely run in the browser (or any environment). You will use:

- **Hono** – a modern, lightweight web framework for building APIs (similar to Express, but optimized for Edge environments).
- **esbuild** – an extremely fast JavaScript bundler and minifier, which also offers a `transform` API for converting modern JavaScript into browser-friendly code.

Your Task

Build a web server using **Hono** with a POST endpoint:

```
POST /transform
```

This endpoint should:

1. Accept a JSON payload with:
   - `code`: A string of JavaScript code provided by the user.
   - `target`: A string representing the target ECMAScript output (e.g., `"es2017"`, `"es2020"`, etc.).
   - `loader`: A string representing the source code type (e.g., `"js"`, `"ts"`).
2. Use esbuild's transform API to transpile the provided script using the given `target` and `loader`. Validate that `target` and `loader` are valid esbuild options.
3. Return the transformed (stringified) code in a JSON response.
4. Ensure the transformation does **not allow any potentially unsafe browser behavior** (like eval, top-level await, or `import()` statements).

Example Request/Response:

```
POST /transform
Content-Type: application/json
{
  "code": "const x = 12; const y = 13; console.log(x)",
  "target": "es2015",
  "loader": "ts"
}
```

```
{
  "code": "\"use strict\";\nconsole.log(12);\n",
  "map": "{\n  \"version\": 3,\n  \"sources\": [\"<stdin>\"],\n  \"sourcesContent\": [\"const x = 12; const y = 13; console.log(x)\"],\n  \"mappings\": \";AAA4B,QAAQ,IAAI,EAAC\",\n  \"names\": []\n}\n"
}
```


Bonus / Nice to Haves:

- Add basic input validation with a schema validator like Zod.
- Add support for returning **warnings** from esbuild (e.g., use of deprecated features).
- Use esbuild transform options like define, platform, supported for better security. Explore other options as well.
- Add rate limiter.
- Deploy your Hono server on Cloudflare Workers or Deno Deploy.

Resources:

- Hono: Hono - Web framework built on Web Standards
- esbuild `transform`: esbuild - API
- Valid `target` values: esbuild - API
- Valid `loader` values: esbuild - API
- Zod (optional): TypeScript-first schema validation with static type inference
- Zod OpenAPI Hono Integration: middleware/packages/zod-openapi at main · honojs/middleware